

Uebungsblatt 3

„Mustererkennung“

J. Cavojska, N. Lehmann, R. Toudic

05.05.2015

1 Aufbereitung der Daten

```
1 % Trainingsdaten, Testdaten und Clusterdaten laden
2 A = load('pendigits-training.txt');
3 B = load('pendigits-testing.txt');
4 C = load('clusters.txt');
5
6 %Dimensionen der Trainingsdaten
7 A_n = size(A,2);
8 A_m = size(A,1);
9
10 % Dimensionen der Testdaten
11 B_n = size(B,2);
12 B_m = size(B,1);
13
14 % Daten ohne die Zugliniennummer (Trainings- und Testdaten)
15 A_nl = A(:,1:A_n -1);
16 B_nl = B(:,1:B_n -1);
17
18 % Trainingsdaten aufgeteilt nach Zugliniennummer
19 A_0 = A((A(:,17)==0),:);
20 A_1 = A((A(:,17)==1),:);
21 A_2 = A((A(:,17)==2),:);
22 A_3 = A((A(:,17)==3),:);
23 A_4 = A((A(:,17)==4),:);
24 A_5 = A((A(:,17)==5),:);
25 A_6 = A((A(:,17)==6),:);
26 A_7 = A((A(:,17)==7),:);
27 A_8 = A((A(:,17)==8),:);
28 A_9 = A((A(:,17)==9),:);
29
30 % Trainingsdaten aufgeteilt nach Zugliniennummer ohne Zugliniennummer
31 A_0_nl = A_0(:,1:A_n -1);
32 A_1_nl = A_1(:,1:A_n -1);
33 A_2_nl = A_2(:,1:A_n -1);
```

```

34 A_3_n1 = A_3(:,1:A_n -1);
35 A_4_n1 = A_4(:,1:A_n -1);
36 A_5_n1 = A_5(:,1:A_n -1);
37 A_6_n1 = A_6(:,1:A_n -1);
38 A_7_n1 = A_7(:,1:A_n -1);
39 A_8_n1 = A_8(:,1:A_n -1);
40 A_9_n1 = A_9(:,1:A_n -1);

```

2 Aufgabe 1 (Multivariate Normalverteilung)

Laden Sie die Dateien *pendigits-testing.txt* und *pendigitstraining.txt*. Jede Zeile dieser Dateien ist ein Datensatz für einen Linienzug einer Ziffer bestehend aus 17 Zahlen, die durch Leerzeichen getrennt sind. Die ersten 16 Zahlen sind 8 X/Y-Koordinatenpaare. Die letzte Zahl ist die Ziffer, die der Linienzug darstellen soll.

Berechnen Sie die multivariate (mehrdimensionale) Normalverteilung (Erwartungswert und Kovarianzmatrix) für den 16-dimensionalen Koordinatenvektor jeweils für alle 10 Ziffern anhand der Werte aus *pendigitstraining.txt*.

```

1 % Erwartungswert fuer jede Koordinate fuer jeden Zug (0 bis 9)
2 E_A_0 = mean(A_0_n1);
3 E_A_1 = mean(A_1_n1);
4 E_A_2 = mean(A_2_n1);
5 E_A_3 = mean(A_3_n1);
6 E_A_4 = mean(A_4_n1);
7 E_A_5 = mean(A_5_n1);
8 E_A_6 = mean(A_6_n1);
9 E_A_7 = mean(A_7_n1);
10 E_A_8 = mean(A_8_n1);
11 E_A_9 = mean(A_9_n1);
12
13 % Kovarianzmatrix fuer jeden Zug (0 bis 9)
14 CVM_A_0 = cov(A_0_n1);
15 CVM_A_1 = cov(A_1_n1);
16 CVM_A_2 = cov(A_2_n1);
17 CVM_A_3 = cov(A_3_n1);
18 CVM_A_4 = cov(A_4_n1);
19 CVM_A_5 = cov(A_5_n1);
20 CVM_A_6 = cov(A_6_n1);
21 CVM_A_7 = cov(A_7_n1);
22 CVM_A_8 = cov(A_8_n1);
23 CVM_A_9 = cov(A_9_n1);
24
25 % Multivariate PDF generieren fuer jeden Zug (0 bis 9)
26 A_0_mvpdf = mvnpdf(A_0_n1, E_A_0, CVM_A_0);

```

```

27 A_1_mvpdf = mvnpdf(A_1_n1, E_A_1, CVM_A_1);
28 A_2_mvpdf = mvnpdf(A_2_n1, E_A_2, CVM_A_2);
29 A_3_mvpdf = mvnpdf(A_3_n1, E_A_3, CVM_A_3);
30 A_4_mvpdf = mvnpdf(A_4_n1, E_A_4, CVM_A_4);
31 A_5_mvpdf = mvnpdf(A_5_n1, E_A_5, CVM_A_5);
32 A_6_mvpdf = mvnpdf(A_6_n1, E_A_6, CVM_A_6);
33 A_7_mvpdf = mvnpdf(A_7_n1, E_A_7, CVM_A_7);
34 A_8_mvpdf = mvnpdf(A_8_n1, E_A_8, CVM_A_8);
35 A_9_mvpdf = mvnpdf(A_9_n1, E_A_9, CVM_A_9);
36
37 % A-Priori-Wahrscheinlichkeit fuer jeden Zug (0 bis 9)
38 A_x_apriori = 1 / length(unique(A(:,A_n)));
39
40 % A-Posteriori-Wahrscheinlichkeit fuer jeden Zug (0 bis 9)
41 A_0_aposteriori = A_0_mvpdf * A_x_apriori;
42 A_1_aposteriori = A_1_mvpdf * A_x_apriori;
43 A_2_aposteriori = A_2_mvpdf * A_x_apriori;
44 A_3_aposteriori = A_3_mvpdf * A_x_apriori;
45 A_4_aposteriori = A_4_mvpdf * A_x_apriori;
46 A_5_aposteriori = A_5_mvpdf * A_x_apriori;
47 A_6_aposteriori = A_6_mvpdf * A_x_apriori;
48 A_7_aposteriori = A_7_mvpdf * A_x_apriori;
49 A_8_aposteriori = A_8_mvpdf * A_x_apriori;
50 A_9_aposteriori = A_9_mvpdf * A_x_apriori;

```

Klassifizieren Sie die Ziffern in pendigitstesting.txt anhand der entsprechenden A-posteriori Wahrscheinlichkeitsdichtefunktionen. Nehmen dabei Sie eine gleichverteilte Apriori Wahrscheinlichkeit fr jede Ziffer an.

```

1 % Klassifizierung der Testdaten (Metrik: L2-Norm)
2 M_classify = [];
3 for index = 1:size(B,1)
4     testData = B(index,1:B_n -1);
5
6     % multivariate PDF f r Testdatensatz (fuer jede Zuglinie)
7     A_0_aposteriori_predict = mvnpdf(testData, E_A_0, CVM_A_0);
8     A_1_aposteriori_predict = mvnpdf(testData, E_A_1, CVM_A_1);
9     A_2_aposteriori_predict = mvnpdf(testData, E_A_2, CVM_A_2);
10    A_3_aposteriori_predict = mvnpdf(testData, E_A_3, CVM_A_3);
11    A_4_aposteriori_predict = mvnpdf(testData, E_A_4, CVM_A_4);
12    A_5_aposteriori_predict = mvnpdf(testData, E_A_5, CVM_A_5);
13    A_6_aposteriori_predict = mvnpdf(testData, E_A_6, CVM_A_6);
14    A_7_aposteriori_predict = mvnpdf(testData, E_A_7, CVM_A_7);
15    A_8_aposteriori_predict = mvnpdf(testData, E_A_8, CVM_A_8);
16    A_9_aposteriori_predict = mvnpdf(testData, E_A_9, CVM_A_9);
17
18    % L2 Norm der aposteriori Vorhersage
19    A0_l2 = norm(A_0_aposteriori_predict);
20    A1_l2 = norm(A_1_aposteriori_predict);
21    A2_l2 = norm(A_2_aposteriori_predict);
22    A3_l2 = norm(A_3_aposteriori_predict);
23    A4_l2 = norm(A_4_aposteriori_predict);
24    A5_l2 = norm(A_5_aposteriori_predict);
25    A6_l2 = norm(A_6_aposteriori_predict);

```

```

26     A7_l2 = norm(A_7_aposteriori_predict);
27     A8_l2 = norm(A_8_aposteriori_predict);
28     A9_l2 = norm(A_9_aposteriori_predict);
29
30     % Bestimmung des Maximums (aposteriori Vorhersage)
31     [maxValue, indexAtMaxValue] = max([A0_l2, A1_l2, A2_l2, A3_l2, A4_l2, ←
        A5_l2, A6_l2, A7_l2, A8_l2, A9_l2]);
32
33     % Bayes Klassifikation (Welche aposteriori Vorhersage war die Groesste?)
34     if (maxValue == A0_l2) % train 0 predicted
35         tmpVector = [B(index,1:B_n -1),B(index,B_n),0];
36         M_classify = vertcat(M_classify,tmpVector);
37
38     elseif (maxValue == A1_l2) % train 1 predicted
39         tmpVector = [B(index,1:B_n -1),B(index,B_n),1];
40         M_classify = vertcat(M_classify,tmpVector);
41
42     elseif (maxValue == A2_l2) % train 2 predicted
43         tmpVector = [B(index,1:B_n -1),B(index,B_n),2];
44         M_classify = vertcat(M_classify,tmpVector);
45
46     elseif (maxValue == A3_l2) % train 3 predicted
47         tmpVector = [B(index,1:B_n -1),B(index,B_n),3];
48         M_classify = vertcat(M_classify,tmpVector);
49
50     elseif (maxValue == A4_l2) % train 4 predicted
51         tmpVector = [B(index,1:B_n -1),B(index,B_n),4];
52         M_classify = vertcat(M_classify,tmpVector);
53
54     elseif (maxValue == A5_l2) % train 5 predicted
55         tmpVector = [B(index,1:B_n -1),B(index,B_n),5];
56         M_classify = vertcat(M_classify,tmpVector);
57
58     elseif (maxValue == A6_l2) % train 6 predicted
59         tmpVector = [B(index,1:B_n -1),B(index,B_n),6];
60         M_classify = vertcat(M_classify,tmpVector);
61
62     elseif (maxValue == A7_l2) % train 7 predicted
63         tmpVector = [B(index,1:B_n -1),B(index,B_n),7];
64         M_classify = vertcat(M_classify,tmpVector);
65
66     elseif (maxValue == A8_l2) % train 8 predicted
67         tmpVector = [B(index,1:B_n -1),B(index,B_n),8];
68         M_classify = vertcat(M_classify,tmpVector);
69
70     else % train 9 predicted
71         tmpVector = [B(index,1:B_n -1),B(index,B_n),9];
72         M_classify = vertcat(M_classify,tmpVector);
73
74     end % end-if
75
76 end % end-for_each

```

Geben Sie die die Konfusionsmatrix und Klassifikationsgte aus.

```

1 % Konfusionsmatrix (Rows: actual classes, Columns: predicted classes)
2 % 341      0      0      0      0      0      0      0      22      0
3 %      0    350     12      0      1      0      0      0      1      0
4 %      0      8    355      0      0      0      0      1      0      0
5 %      0      9      0    320      0      1      0      1      0      5
6 %      0      0      0      0    362      0      0      0      0      2
7 %      0      0      0      1      0    323      0      0      2      9
8 %      0      0      0      0      0      0    325      0     11      0
9 %      0     28      0      0      0      0      0    314      5     17
10 %      0      0      0      0      0      0      0      0    336      0
11 %      0      5      0      0      0      0      0      1      1    329
12 knownClass = M_classify(:, B_n);
13 predictedClass = M_classify(:, B_n + 1);
14 confusion_matrix = confusionmat(knownClass, predictedClass)
15
16 % Klassifikationsguete = 0.9591
17 M_m = size(M_classify, 1);
18 corret_predicted = 0;
19 for index = 1:M_m
20     if M_classify(index, B_n) == M_classify(index, B_n + 1)
21         corret_predicted = corret_predicted + 1;
22     end
23 end
24 classification_quality = corret_predicted / M_m

```

3 Aufgabe 2 (Multivariate Normalverteilung mit PCA)

a) Geben sie die erste Hauptkomponente der Daten in *pendigitstraining.txt* an.

```

1 % Kovarianzmatrix
2 CVM_A = cov(A_n1); % zentriert durch cov()
3 CVM_B = cov(B_n1); % zentriert durch cov()
4
5 % Eigenvektoren (VB) und Eigenwerte (DB) der Kovarianzmatrix (balanciert)
6 [VB,DB] = eig(CVM_A);
7 EigVec_CVM_A = VB; % Eigenvektoren von CVM_A
8 EigVal_CVM_A = DB; % Diagonalmatrix der Eigenwerte zu CVM_A
9
10 [VB,DB] = eig(CVM_B);
11 EigVec_CVM_B = VB; % Eigenvektoren von CVM_B
12 EigVal_CVM_B = DB; % Diagonalmatrix der Eigenwerte zu CVM_B
13
14 X = EigVec_CVM_A(:, [16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]);
15

```

```

16 % get the principal component (the eigenvector with the highest eigenvalue):
17 % the eigenvalues in EigVal_CVM_A are already sorted (ascending), so we can ←
    just get the last column:
18 first_principal_component = EigVec_CVM_A(:,end)
19
20 % erste Hauptkomponente:
21 % 0.0713
22 % 0.0722
23 % -0.2017
24 % -0.1531
25 % -0.2704
26 % -0.3593
27 % -0.1578
28 % -0.4137
29 % -0.1183
30 % -0.1779
31 % -0.0376
32 % 0.2106
33 % 0.0705
34 % 0.4627
35 % 0.0877
36 % 0.4574

```

b) Reduzieren Sie die Dimension des pendigits-Datensatzes mittels einer Hauptkomponentenanalyse (PCA) und klassifizieren die Testdaten anhand der Trainingsdaten mit einem BayesKlassifikator (wie Aufgabe 1).

```

1  for dim = [1:16]
2
3      % Unterraum erzeugen
4      pca_ur = X(:,1:dim);
5
6      % Abbildung der Trainingsdaten auf Unterraum
7      A_0_ur = A_0_n1 * pca_ur; % Datenpunkte fuer Zuglinie 0
8      A_1_ur = A_1_n1 * pca_ur; % Datenpunkte fuer Zuglinie 1
9      A_2_ur = A_2_n1 * pca_ur; % Datenpunkte fuer Zuglinie 2
10     A_3_ur = A_3_n1 * pca_ur; % Datenpunkte fuer Zuglinie 3
11     A_4_ur = A_4_n1 * pca_ur; % Datenpunkte fuer Zuglinie 4
12     A_5_ur = A_5_n1 * pca_ur; % Datenpunkte fuer Zuglinie 5
13     A_6_ur = A_6_n1 * pca_ur; % Datenpunkte fuer Zuglinie 6
14     A_7_ur = A_7_n1 * pca_ur; % Datenpunkte fuer Zuglinie 7
15     A_8_ur = A_8_n1 * pca_ur; % Datenpunkte fuer Zuglinie 8
16     A_9_ur = A_9_n1 * pca_ur; % Datenpunkte fuer Zuglinie 9
17
18     % Abbildung der Testdaten auf Unterraum
19     B_ur = B_n1 * pca_ur;
20
21     % Erwartungswerte bestimmen
22     E_A_0_ur = mean(A_0_ur);
23     E_A_1_ur = mean(A_1_ur);
24     E_A_2_ur = mean(A_2_ur);
25     E_A_3_ur = mean(A_3_ur);
26     E_A_4_ur = mean(A_4_ur);
27     E_A_5_ur = mean(A_5_ur);

```

```

28     E_A_6_ur = mean(A_6_ur);
29     E_A_7_ur = mean(A_7_ur);
30     E_A_8_ur = mean(A_8_ur);
31     E_A_9_ur = mean(A_9_ur);
32
33     % Kovarianzmatrizen bestimmen
34     CVM_A_0_ur = cov(A_0_ur);
35     CVM_A_1_ur = cov(A_1_ur);
36     CVM_A_2_ur = cov(A_2_ur);
37     CVM_A_3_ur = cov(A_3_ur);
38     CVM_A_4_ur = cov(A_4_ur);
39     CVM_A_5_ur = cov(A_5_ur);
40     CVM_A_6_ur = cov(A_6_ur);
41     CVM_A_7_ur = cov(A_7_ur);
42     CVM_A_8_ur = cov(A_8_ur);
43     CVM_A_9_ur = cov(A_9_ur);
44
45     % Klassifizierung der Testdaten (Metrik: L2-Norm)
46     M_classify = [];
47     for index = 1:size(B_ur,1)
48         testData = B_ur(index,:);
49
50         % multivariate PDF fuer Testdatensatz ( f r jede Zuglinie)
51         A_0_aposteriori_predict = mvnpdf(testData, E_A_0_ur, CVM_A_0_ur) * ←
            A_x_apriori;
52         A_1_aposteriori_predict = mvnpdf(testData, E_A_1_ur, CVM_A_1_ur) * ←
            A_x_apriori;
53         A_2_aposteriori_predict = mvnpdf(testData, E_A_2_ur, CVM_A_2_ur) * ←
            A_x_apriori;
54         A_3_aposteriori_predict = mvnpdf(testData, E_A_3_ur, CVM_A_3_ur) * ←
            A_x_apriori;
55         A_4_aposteriori_predict = mvnpdf(testData, E_A_4_ur, CVM_A_4_ur) * ←
            A_x_apriori;
56         A_5_aposteriori_predict = mvnpdf(testData, E_A_5_ur, CVM_A_5_ur) * ←
            A_x_apriori;
57         A_6_aposteriori_predict = mvnpdf(testData, E_A_6_ur, CVM_A_6_ur) * ←
            A_x_apriori;
58         A_7_aposteriori_predict = mvnpdf(testData, E_A_7_ur, CVM_A_7_ur) * ←
            A_x_apriori;
59         A_8_aposteriori_predict = mvnpdf(testData, E_A_8_ur, CVM_A_8_ur) * ←
            A_x_apriori;
60         A_9_aposteriori_predict = mvnpdf(testData, E_A_9_ur, CVM_A_9_ur) * ←
            A_x_apriori;
61
62         % L2 Norm der aposteriori Vorhersage
63         A0_12 = norm(A_0_aposteriori_predict);
64         A1_12 = norm(A_1_aposteriori_predict);
65         A2_12 = norm(A_2_aposteriori_predict);
66         A3_12 = norm(A_3_aposteriori_predict);
67         A4_12 = norm(A_4_aposteriori_predict);
68         A5_12 = norm(A_5_aposteriori_predict);
69         A6_12 = norm(A_6_aposteriori_predict);
70         A7_12 = norm(A_7_aposteriori_predict);
71         A8_12 = norm(A_8_aposteriori_predict);
72         A9_12 = norm(A_9_aposteriori_predict);
73
74         % Bestimmung des Maximums (aposteriori Vorhersage)

```

```

75     [maxValue, indexAtMaxValue] = max([A0_12, A1_12, A2_12, A3_12, A4_12, ←
76         , A5_12, A6_12, A7_12, A8_12, A9_12]);
77
78     % Bayes Klassifikation (Welche aposteriori Vorhersage war die ←
79     Groesste?)
80     if (maxValue == A0_12) % train 0 predicted
81         tmpVector = [B_ur(index,:), B(index, B_n), 0];
82         M_classify = vertcat(M_classify, tmpVector);
83     elseif (maxValue == A1_12) % train 1 predicted
84         tmpVector = [B_ur(index,:), B(index, B_n), 1];
85         M_classify = vertcat(M_classify, tmpVector);
86     elseif (maxValue == A2_12) % train 2 predicted
87         tmpVector = [B_ur(index,:), B(index, B_n), 2];
88         M_classify = vertcat(M_classify, tmpVector);
89     elseif (maxValue == A3_12) % train 3 predicted
90         tmpVector = [B_ur(index,:), B(index, B_n), 3];
91         M_classify = vertcat(M_classify, tmpVector);
92     elseif (maxValue == A4_12) % train 4 predicted
93         tmpVector = [B_ur(index,:), B(index, B_n), 4];
94         M_classify = vertcat(M_classify, tmpVector);
95     elseif (maxValue == A5_12) % train 5 predicted
96         tmpVector = [B_ur(index,:), B(index, B_n), 5];
97         M_classify = vertcat(M_classify, tmpVector);
98     elseif (maxValue == A6_12) % train 6 predicted
99         tmpVector = [B_ur(index,:), B(index, B_n), 6];
100        M_classify = vertcat(M_classify, tmpVector);
101    elseif (maxValue == A7_12) % train 7 predicted
102        tmpVector = [B_ur(index,:), B(index, B_n), 7];
103        M_classify = vertcat(M_classify, tmpVector);
104    elseif (maxValue == A8_12) % train 8 predicted
105        tmpVector = [B_ur(index,:), B(index, B_n), 8];
106        M_classify = vertcat(M_classify, tmpVector);
107    else % train 9 predicted
108        tmpVector = [B_ur(index,:), B(index, B_n), 9];
109        M_classify = vertcat(M_classify, tmpVector);
110    end % end-if
111    end % end-for_each
112
113    M_classify_n = size(M_classify, 2);
114    M_classify_m = size(M_classify, 1);
115
116    % Konfusionsmatrix
117    knownClass = M_classify(:, M_classify_n - 1);
118    predictedClass = M_classify(:, M_classify_n);
119    disp(['Number of dimensions: ', num2str(dim)]);
120    confusionmatrix = confusionmat(knownClass, predictedClass)
121
122    % Klassifikationsguete
123    corret_predicted = 0;
124    for index = 1:M_classify_m
125        if M_classify(index, M_classify_n - 1) == M_classify(index, ←
126            M_classify_n)
127            corret_predicted = corret_predicted + 1;
128        end
129    end
130    classification_quality = corret_predicted / M_classify_m

```



```
129 end % for dim
```

Geben Sie die Klassifikationsgüte für jede der Dimensionen von 1 bis 15 aus.

```
1 Number of dimensions: 1
2 classification_quality = 0.4042
3
4 Number of dimensions: 2
5 classification_quality = 0.6515
6
7 Number of dimensions: 3
8 classification_quality = 0.7882
9
10 Number of dimensions: 4
11 classification_quality = 0.8382
12
13 Number of dimensions: 5
14 classification_quality = 0.8708
15
16 Number of dimensions: 6
17 classification_quality = 0.8957
18
19 Number of dimensions: 7
20 classification_quality = 0.9062
21
22 Number of dimensions: 8
23 classification_quality = 0.9260
24
25 Number of dimensions: 9
26 classification_quality = 0.9491
27
28 Number of dimensions: 10
29 classification_quality = 0.9480
30
31 Number of dimensions: 11
32 classification_quality = 0.9537
33
34 Number of dimensions: 12
35 classification_quality = 0.9540
36
37 Number of dimensions: 13
38 classification_quality = 0.9554
39
40 Number of dimensions: 14
41 classification_quality = 0.9565
42
43 Number of dimensions: 15
44 classification_quality = 0.9594
45
46 Number of dimensions: 16
47 classification_quality = 0.9591
```

4 Aufgabe 3 (k-Means)

Laden Sie die Datei `clusters.txt`. Jede Zeile dieser Datei entspricht einem X/Y Koordinatenpaar.

Clustern Sie den Datensatz mit dem k -Means-Algorithmus.

Visualisieren Sie die Clusterzentren und Zuordnung der Punkte der ersten 5 Iterationsschritte mit $k=3$ (Also insgesamt 5 Bilder)

```
1 C = load('clusters.txt');
2 k = 3;
3 numIterations = 5;
4
5 mean1 = C(1,:); % mean1, selected randomly
6 mean2 = C(2,:); % mean2, selected randomly
7 mean3 = C(3,:); % mean3, selected randomly
8 mean1_elems = []; % elements belonging to mean1
9 mean2_elems = []; % elements belonging to mean2
10 mean3_elems = []; % elements belonging to mean3
11 plotArray = [];
12
13 for iter=1:numIterations
14     mean1_elems = [];
15     mean2_elems = [];
16     mean3_elems = [];
17     for elem=1:size(C,1) % iterate over all elements
18         dist = sqrt(abs(C(elem,1) - mean1(:,1))^2 + abs(C(elem,2) - mean1(:,2))^2);
19         closest = mean1;
20         dist2 = sqrt(abs(C(elem,1) - mean2(:,1))^2 + abs(C(elem,2) - mean2(:,2))^2);
21         if dist > dist2
22             closest = mean2;
23             dist = dist2;
24         end
25         dist3 = sqrt(abs(C(elem,1) - mean3(:,1))^2 + abs(C(elem,2) - mean3(:,2))^2);
26         if dist > dist3
27             closest = mean3;
28             dist = dist3;
29         end
30         if closest == mean1
31             mean1_elems = vertcat(mean1_elems, C(elem, :));
32         elseif closest == mean2
33             mean2_elems = vertcat(mean2_elems, C(elem, :));
34         else
35             mean3_elems = vertcat(mean3_elems, C(elem, :));
36         end
37     end
38     mean1_elems;
39     mean2_elems;
40     mean3_elems;
41
42     % Visualisierung der Clusterzentren
```

```

43     plotOfIteration = 1; % which iteration do we want to see a plot for?
44     if iter == plotOfIteration
45         % x = min(mean1_elems):max(mean1_elems)
46         mean1_elems_x = mean1_elems(:,1); % x coordinates of all elements ←
            belonging to mean1
47         mean1_elems_y = mean1_elems(:,2); % y coordinates of all elements ←
            belonging to mean1
48         mean2_elems_x = mean2_elems(:,1);
49         mean2_elems_y = mean2_elems(:,2);
50         mean3_elems_x = mean3_elems(:,1);
51         mean3_elems_y = mean3_elems(:,2);
52         scatter(mean1_elems_x, mean1_elems_y, 40, [1 0 0])
53         hold on
54         scatter(mean1(:,1), mean1(:,2), 60, [.3 0 0], 'filled')
55         hold on
56         scatter(mean2_elems_x, mean2_elems_y, 40, [0 1 0])
57         hold on
58         scatter(mean2(:,1), mean2(:,2), 60, [0 .3 0], 'filled')
59         hold on
60         scatter(mean3_elems_x, mean3_elems_y, 40, [0 0 1])
61         hold on
62         scatter(mean3(:,1), mean3(:,2), 60, [0 0 .3], 'filled')
63     end
64
65     % Berechnung der neuen Clusterzentren aus den berechneten Cluster←
        Datenpunkten
66     mean1 = [mean(mean1_elems(:,1)), mean(mean1_elems(:,2))];
67     mean2 = [mean(mean2_elems(:,1)), mean(mean2_elems(:,2))];
68     mean3 = [mean(mean3_elems(:,1)), mean(mean3_elems(:,2))];
69 end

```

4.1 Grafiken zu den ersten 5 k-Means-Iterationen:

